Multi-objective Algorithms for the Single Machine Scheduling Problem with Sequence-dependent Family Setups

Marcelo Ferreira Rego[1], Marcone Jamilson Freitas Souza[1], Igor Machado Coelho[2], and José Elias Claudio Arroyo[3]

[1] Departament of Computing – Federal University of Ouro Preto
Ouro Preto, MG, Brazil – 35400-000
marcelofr@gmail.com, marcone@iceb.ufop.br
[2] Institute of Computing – Fluminense Federal University
Niterói, RJ, Brazil – 24210-240
imcoelho@ic.uff.br
[3] Departament of Computing – Federal University of Viçosa
Viçosa, MG, Brazil – 36571-000
jarroyo@dpi.ufv.br

Abstract. This work treats the single machine scheduling problem in which the setup time depends on the sequence and the job family. The objective is to minimize the makespan and the total weighted tardiness. In order to solve the problem two multi-objective algorithms are analyzed: one based on Multi-objective Variable Neighborhood Search (`MOVNS`) and another on Pareto Iterated Local Search (`PILS`). Two literature algorithms based on `MOVNS` are adapted to solve the problem, resulting in the `MOVNS_Ottoni` and `MOVNS_Arroyo` variants. Also, a new perturbation procedure for the `PILS` is proposed, yielding the `PILS1` variant. Computational experiments done over randomly generated instances show that `PILS1` is statistically better than all other algorithms in relation to the cardinality, average distance, maximum distance, difference of hypervolume and epsilon metrics.

Keywords–single machine scheduling; multi-objective optimization; pareto iterated local search, multi-objective variable neighborhood search

1 Introduction

Scheduling problems have been extensively studied in the literature. This fact is due to at least two aspects. The first one is the practical interest, since there are various applications on this class of problems in industry field, as for example, textile [11], electronics [15] and iron [21]. The other aspect that interests the study of this kind of problem is the theoretical interest, once most of the scheduling problems belong to the class of $\mathcal{NP}$-hard problems [3].

Although the problem of scheduling jobs involves various objectives, in most of the researches in this field only one objective is considered. When more than

one is considered, usually it is defined only one objective represented by the linear combination of involved objectives, thus, the problem is treated normally in a single-objective approach.

This work discusses the scheduling problem in single machines, in which the setup time of the machine depends on the scheduling and family of the jobs. The grouping of the jobs in the family occurs, for example, in the iron field. In [4], the author shows a process of manufacturing iron products (corner, rebar, bar, etc) in the lamination sector, in which jobs are grouped in families in accordance with the similarity of the products. In this case, the products from same family are those which differ between themselves by the thickness. On those circumstances the setup time is so short and unimportant when compared to the processing time of jobs that is usual to consider it equivalent to zero. The advantage of making this grouping, thus, is that the jobs which belong to the same family, when processed sequentially, do not need setup time.

The problem in hand takes two objectives into account: makespan and total weighted tardiness minimization. It means that instead of looking for a solution which satisfies one or other objective separately, the main goal is to obtain a set of non-dominated solutions, this way each solution that belongs to this set is not worse than any other, considering both objectives simultaneously.

Noticing the computational complexity of the scheduling problems, the most used methods to solve them are metaheuristics. Reviews in literature show that methods inspired by the process of natural evolution, such as Non-dominated Sorting Genetic Algorithm II – `NSGA-II` [7] and Strength Pareto Approach – `SPEA2` [25] are among the most used when multi-objective optimization is concerned. On the other hand, recently it were discovered reports of successful applications of multi-objective methods based on local search, such as Multi-objective Variable Neighborhood Search – `MOVNS` [9] and Pareto Iterated Local Search – `PILS` [10], as shown in the works of [1], [20] and [18]. In this last, for example, it is observed a superiority of `PILS` over `SPEA2`.

Due to the good performance of the algorithms in similar problems, in this work the multi-objective algorithms `MOVNS` and `PILS` are tested to solve the scheduling problem at hand. The `MOVNS` algorithms of [1] and [20] were adapted to solve the problem, giving birth to the `MOVNS_Ottoni` and `MOVNS_Arroyo` variants, respectively. Furthermore, a new perturbation procedure is proposed for `PILS`, giving birth to the `PILS1` variant. Computational experiments showed that the last algorithm outperforms the other tested algorithms.

The rest of this work is organized as follows. In Section 2 the problem characteristics are described. Section 3 presents the proposed multi-objective algorithms, and in Section 4 the used test instances are described as well as the metrics used to assess and compare the developed algorithms. Also in Section 4, the results of the accomplished experiments are presented and analyzed. Section 5 concludes the work.

2 Problem Description

The problem in focus can be defined as follows: there is a set $J = \{1, 2, 3, \ldots, n\}$ with n jobs that have to be scheduled in a single machine at the starting point zero. Each job $j \in J$ has a processing time p_j, a due date d_j and a weight for tardiness β_j. The jobs are grouped in families f according to their characteristics and each family i has n_i jobs. A setup time s_{ik} is required between the execution of two consecutive jobs of different families i and k and, if they are from the same family, no setup time is necessary. Given a sequence π, for each job j a tardiness T_j is associated. Since C_j is the completion time of the job j, its tardiness is calculated by Eq. (1):

$$T_j = \max\{C_j - d_j, 0\} \tag{1}$$

The objectives of the problem in focus are to minimize the makespan $f_1(\pi)$ and the total weighted tardiness $f_2(\pi)$, simultaneously. The values of $f_1(\pi)$ and $f_2(\pi)$ are calculated by Eqs. (2) and (3):

$$f_1(\pi) = \max_{1 \leqslant j \leqslant n} \{C_j\} \tag{2}$$

$$f_2(\pi) = \sum_{1 \leqslant j \leqslant n} \beta_j T_j \tag{3}$$

3 Methodology

A solution is represented by a sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_k, \ldots, \pi_n\}$ in which π_k indicates the k-th job to be done.

The proposed algorithms begin with a set of non-dominated solutions generated by four different heuristics based on the following dispatching rules [22]: Earliest Due Date (`EDD`) rule, generating a scheduling of jobs in a non-decreasing order of their due dates; Shortest Processing Time (`SPT`) rule, generating a scheduling of jobs in a non-decreasing order of their processing time; Longest Processing Time (`LPT`) rule, generating a scheduling of jobs in a non-increasing order of their processing time; Minimum Slack Time (`MST`) rule, generating a scheduling of jobs in a non-decreasing order of the difference between the due date and processing time.

In order to explore the solution space of the problem, insertion and exchange movements are applied changing the scheduling of the jobs, as follows.

Given a sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, the insertion move of a job π_x consists in moving this job to a position y ($y \neq x$ e $y \neq x - 1$). The group of insertion movements in a sequence π defines the neighborhood $N^I(\pi)$ which is composed by $(n-1)^2$ solutions.

Given a sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, the exchanging move between two jobs π_x and π_y consists in moving the job π_x to the position y and the job π_y to the position x. The group of exchanging movements in a sequence π defines the neighborhood $N^T(\pi)$, formed by $\dfrac{n \cdot (n-1)}{2}$ solutions.

3.1 Algorithms based on MOVNS

The Multi-objective Variable Neighborhood Search (MOVNS) is an optimization multi-objective algorithm proposed in [9] and the metaheuristic Variable Neighborhood Search – VNS [19].

Variants of MOVNS Algorithm In literature there are two variants of the MOVNS algorithm. The first one, named as MOVNS_Ottoni, was proposed by [20] and consists in adding an intensification procedure to the original MOVNS. The second variant of MOVNS, named MOVNS_Arroyo, was proposed by [1] and consists in adding another different intensification.

3.2 Algorithms based on PILS

Pareto Iterated Local Search – PILS is an optimization multi-objective algorithm proposed by [10], with a structure based on the meta-heuristic Iterated Local Search – ILS [17]. PILS basic pseudo-code is presented in Algorithm 1.

In Algorithm 1, initially it is obtained a set of non-dominated solutions (ND) (line 1), using four different heuristics described previously. After this, one of the solutions of the set ND is selected randomly (line 2). In each iteration of the external loop (lines 3 – 27) all neighbors of the current solution are explored (lines 5 – 17). If a neighbor solution dominates the current solution, then this neighbor solution becomes the new current solution, the neighborhoods are randomly reordered and the procedure returns to the first neighborhood of the new generated order. This procedure is repeated while there are non-visited solutions in set ND. After all solutions of set ND are visited – when the algorithm is in an local optimum concerning the explored neighborhood – is a solution is randomly selected from set ND (line 23) and a perturbation is applied (line 24), as described as follows. After this, all neighborhood of the current solution is explored (lines 5 – 17). In the case that all neighbors of the solution generated through the perturbation are dominated by any solution of set ND, then the perturbation procedure is repeated. The most external loop (lines 3 – 27) is repeated while the stopping criterion is not met.

A solution is perturbed in order to explore other local optima. The original perturbation strategy of PILS, from [10], works in the following way: initially a solution π from the set ND is randomly selected. Then, a position $j \leq n-4$ and its four consecutively jobs of π are randomly chosen, i.e., positions j, $j+1$, $j+2$ and $j+3$. A perturbed solution s'' is then generated by applying an exchanging move on the jobs in the positions j and $j+3$, as well as the jobs in the positions $j+1$ and $j+2$. This way, the jobs before j and the jobs ahead of $j+3$ are kept in their respective positions after the perturbation application.

In this work, the perturbation procedure of a solution (line 24 from Algorithm 1) has been modified when concerning the proposal of [10]. The perturbation is applied in levels, varying from 1 to $(n/2-1)$. In each level p, $p+1$ modifications are made on the solution. This way, on the lowest pertubation level two exchanges are applied while on the highest level $n/2$ exchanges are made.

Algorithm 1: PILS

```
Input : Non-dominated set ND, k neighborhoods, stopping criterion
ND ← InitialSolution();
Select a solution s ∈ ND ;
while stopping criterion is not satisfied do
    i ← 1 ;
    while i < k ∧ stopping criterion is not satisfied do
        foreach s' ∈ N^i(s) do
            ND ← non-dominated set of ND ∪ {s'} ;
        end
        if ∃s' ∈ N^i(s) | s' dominates s then
            s ← s' ;
            Reorder the neighborhoods N^1, ..., N^k ;
            i ← 1 ;
        end
        else
            i++;
        end
    end
    Mark s as visited ;
    if ∃s' ∈ ND | s' not yet been visited then
        s ← s';
    end
    else
        Select a solution s' ∈ ND ;
        s'' ← Perturbation(s') ;
        s ← s'' ;
    end
end
return ND ;
```

The level p of perturbation increases as the perturbation is not able to generate a non-dominated solution related to the set ND. The increasing is made by adding a unit of value to the current level of perturbation. When a non-dominated solution related to the set ND is found, the perturbation level returns to its lowest value, 1 in this case. If the perturbation level reaches its maximum value – $(n/2-1)$ – and it is still not possible to generate a non-dominated solution in relation to the set ND, the perturbation level returns to its lowest value. The proposed procedure works as follows. A solution π from the set ND is randomly selected. Then it is chosen, also randomly, a subset of consecutive jobs of π on the positions $j, j+1, \ldots, j+2p+1$. Then, exchanges are applied between the pairs of jobs $(j, j+2p+1), (j+1, j+2p), \ldots, (j+p, j+p+1)$. This way, the procedure makes $p+1$ exchanging moves on each call from the perturbation procedure. The `PILS` algorithm modified as such was named `PILS1`.

4 Computational Experiments

All algorithms were coded on C++ language and the tests were done in a Intel® Core™ 2 Quad 2.4 GHz with 6GB RAM.

The stopping criterion of each algorithm is a maximum `CPU` time proportional to the size of the instance. This criterion is common in literature and it has been established as $1000 \times n$ milliseconds, in which n is the number of jobs of the instance. For each instance 30 tests were performed, each one with a different random seed.

4.1 Instances

To assess the algorithms, instances were generated in a random way and with uniform distribution. As in [14], the number of jobs is a integer number $n \in \{60, 80, 100\}$, the number of families $f \in \{2, 3, 4, 5\}$ and the processing time is an integer number in the interval $[1, 99]$. The due date of the jobs was generated as in [2], being defined in the interval $(0, h\sum p_j)$, with $h \in \{0.5; 1.5; 2.5; 3.5\}$. Finally, the setup time between jobs families are integer numbers whose values belong to three classes of intervals: class S $[10, 20]$; class M $[51, 100]$ and class L $[101, 200]$.

The formation of such setup time intervals is a suggestion proposed in [13]. The class S setup time is relatively smaller than the average processing time. The class M setup time is close to the average processing time while the class L setup time is relatively bigger than the average processing time.

By combining the parameters of the number of jobs, number of families, number of intervals to the due dates and the number of classes of intervals to the setup time, an amount of 144 different instances were generated.Note that, for each n, 48 instances were generated.

4.2 Performance Assessment Metrics

The comparison of two sets of non-dominated points, A and B, obtained respectively through two optimization multi-objective algorithms is not a trivial task. Many performance assessment metrics have been proposed on literature [12,23,6,8]. However, these metrics must be chosen in a proper way to make a fair comparison of algorithms.

In this work, five performance assessment metrics are used and they are called: cardinality [12], average distance [5], maximum distance [16], hypervolume [24] and epsilon [8]. In [8] it is shown that the hypervolume and epsilon metrics provide trustworthy measures, especially when two algorithms have similar performances.

The quality of a set of non-dominated points obtained by an algorithm, in a given instance, is assessed in relation to the set composed by all non-dominated points found during all experiments. This is called the set of reference points R.

4.3 Results

The results presented on the following tables were obtained by the developed algorithms with different evaluation metrics. In the first column of these tables is indicated the group n of 48 instances (with number of jobs n). In the other columns the results of each algorithm are presented, with 30 algorithm executions. For each metric the average and best results are presented so as the average of the results in all instances.

Tables 1 presents the average and best results obtained in 30 runs of the algorithms considering the cardinality metric.

Table 1. Cardinality Metric Results

n	Algorithm									
	MOVNS		MOVNS_Ottoni		MOVNS_Arroyo		PILS		PILS1	
	Avg.	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**
60	2.18	12.89	2.79	15.40	3.21	17.24	2.78	10.08	**32.82**	**61.07**
80	0.25	3.41	0.56	8.67	0.54	9.86	0.36	4.61	**11.64**	**50.10**
100	0.07	2.13	0.15	3.97	0.37	8.36	0.15	2.55	**4.40**	**53.87**
Average	0.83	6.15	1.17	9.35	1.37	11.82	1.10	5.74	**16.29**	**55.02**

As can be seen from Table 1, PILS1 algorithm is able to generate a superior number of non-dominated solutions compared to the other algorithms. Besides this, the number of reference solutions generated by the PILS1 algorithm is, in average, at least seven times higher than the one generated by any other algorithm.

Table 2 presents the average and best results in 30 runs of the algorithms considering the average distance metric.

Table 2. Average Distance Metric

n	Algorithm									
	MOVNS		MOVNS_Ottoni		MOVNS_Arroyo		PILS		PILS1	
	Avg.	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**
60	4.18	2.44	3.90	2.22	3.78	2.18	4.72	3.27	**1.20**	**0.25**
80	5.55	2.62	4.92	2.32	4.83	2.27	6.22	3.82	**1.74**	**0.35**
100	7.12	3.34	6.08	2.68	5.92	2.42	7.19	4.14	**2.85**	**0.39**
Average	5.62	2.80	4.97	2.41	4.85	2.29	6.04	3.74	**1.93**	**0.33**

From Table 2 we can conclude that the set of non-dominated solutions produced by the algorithm PILS1 is closer to the reference set than the other algorithms.

Table 3 presents the results obtained by the implemented algorithms considering the maximum distance metric. In this table, both average and best results are presented.

As noticed in Table 3, the set of non-dominated solutions produced by the algorithm PILS1 is in a shorter distance from the reference set.

Table 4 presents the average and best results for the algorithms considering the hypervolume difference metric.

Table 3. Maximum Distance Metric

n	Algorithm									
	MOVNS		MOVNS_Ottoni		MOVNS_Arroyo		PILS		PILS1	
	Avg.	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**
60	10.22	7.57	10.05	7.73	9.86	7.53	11.07	9.38	**5.43**	**1.03**
80	9.16	6.20	8.68	6.01	8.58	5.80	10.00	7.80	**5.23**	**1.05**
100	8.71	4.77	7.85	4.03	7.67	3.87	9.05	5.89	**4.68**	**1.04**
Average	9.36	6.18	8.86	5.92	8.70	5.73	10.04	7.69	**5.11**	**1.04**

Table 4. Hypervolume Difference Metric

n	Algorithm									
	MOVNS		MOVNS_Ottoni		MOVNS_Arroyo		PILS		PILS1	
	Avg.	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**
60	1060.92	556.08	996.74	498.32	969.70	482.68	1205.04	806.87	**267.09**	**14.74**
80	1375.61	660.69	1239.66	568.42	1212.16	534.70	1548.32	968.64	**433.27**	**33.90**
100	1618.76	767.46	1406.73	601.10	1367.94	527.21	1677.25	1015.85	**705.22**	**45.01**
Average	1351.76	661.41	1214.37	555.95	1183.26	514.86	1476.87	930.45	**468.53**	**31.22**

On the Table 4 it is verified that the formed area between the points from the PILS1 algorithm solution set and the points from the non-dominated set are the smallest ones in comparison to the other algorithms. It means that the PILS1 algorithm produces a better cover of the reference set R.

Table 5 presents the average and best results obtained by the algorithms related to the epsilon metric.

Table 5. Epsilon Metric

n	Algorithm									
	MOVNS		MOVNS_Ottoni		MOVNS_Arroyo		PILS		PILS1	
	Avg.	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**	**Avg.**	**Best**
60	1.40	1.21	1.37	1.19	1.36	1.18	1.45	1.30	**1.11**	**1.03**
80	1.20	1.10	1.18	1.08	1.18	1.08	1.21	1.14	**1.07**	**1.02**
100	1.16	1.07	1.13	1.06	1.13	1.05	1.14	1.09	**1.06**	**1.01**
Average	1.25	1.13	1.23	1.11	1.22	1.11	1.27	1.18	**1.08**	**1.02**

From Table 5 it is noticed that the algorithm PILS1 is the one which produces the smallest values to the epsilon metric, indicating that the non-dominated solutions generated by this algorithm are closer to the reference set R.

We also apply the non-parametrical Kruskal-Wallis test in order to verify the statistical superiority of the PILS1 algorithm. According to this test, there is statistical difference between the pairs of algorithms: MOVNS $\times$ PILS1, MOVNS_Ottoni $\times$ PILS1, MOVNS_Arroyo $\times$ PILS1 and PILS $\times$ PILS1.

5 Conclusions

This work dealt with the scheduling problem in single machine where the setup time of the jobs depends on the sequence and on the family, and there are two optimization criteria to be satisfied: makespan and total weight tardiness minimization.

To solve this problem, five multi-objective algorithms based on Pareto Iterated Local Search (PILS) and Multi-objective Variable Neighborhood Search (MOVNS) were implemented. Of these algorithms, three are based on the MOVNS; one of them is the original algorithm and the other two variants of this one found in the literature. And on the other two remaining algorithms, one is the original PILS and the second is a variant proposed in this work, named PILS1. This variant consists in changing the perturbation strategy of PILS.

The algorithms were compared considering Cardinality, Average Distance, Maximum Distance, Hypervolume Difference and Epsilon metrics. The computational results performed in generated instances for the problem were validated by statistical analysis, thus showing that the PILS1 variant is superior to every other algorithm considering the assessed metrics. This way, it is clear the contribution of the perturbation procedure proposed in this work.

Acknowledgments

The authors would like to thank CNPq and FAPEMIG for the financial support on the development of this work.

References

1. J. E. C. Arroyo, R. S. Ottoni, and A. P. Oliveira. Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. *Electronic Notes in Theoretical Computer Science*, 281:5–19, 2011.
2. K. R. Baker and M. J. Magazine. Minimizing maximum lateness with job families. *European Journal of Operational Research*, 127(1):126–139, 2000.
3. P. Brucker. *Scheduling algorithms*. Springer Verlag, 2007.
4. L. M. Bustamante. Minimização do custo de antecipação e atraso para o problema de sequenciamento de uma máquina com tempo de preparação dependente da sequência: aplicação em uma usina siderúrgica. Dissertação de mestrado, Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Minas Gerais, Belo Horizonte, 2007.
5. P. Czyżżak and A. Jaszkiewicz. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47, 1998.
6. K. Deb and S. Jain. Running performance metrics for evolutionary multi-objective optimization. Technical report, 2002. DOI: 10.1.1.9.159.
7. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
8. C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. In *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, volume 216, 2005.
9. M. J. Geiger. Randomised variable neighbourhood search for multi objective optimisation. In *4th EU/ME: Design and Evaluation of Advanced Hybrid Meta-Heuristics*, pages 34–42, 2004.

10. M. J. Geiger. Improvements for multi-objective flow shop scheduling by pareto iterated local search. In *8th Metaheuristics International Conference (MIC)*, pages 195.1 – 195.10, 2009.
11. M. Gendreau, G. Laporte, and E. M. Guimaraes. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 133(1):183–189, 2001.
12. M. P. Hansen and A. Jaszkiewicz. *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1998.
13. A. M. A. Hariri and C. N. Potts. Single machine scheduling with batch set-up times to minimize maximum lateness. *Annals of Operations Research*, 70:75–92, 1997.
14. F. Jin, J. N. D. Gupta, S. Song, and C. Wu. Single machine scheduling with sequence-dependent family setups to minimize maximum lateness. *Journal of the Operational Research Society*, 61(7):1181–1189, 2010.
15. D. W. Kim, K. H. Kim, W. Jang, and F. F. Chen. Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18(3-4):223–231, 2002.
16. J. Knowles and D. Corne. On metrics for comparing nondominated sets. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, volume 1, pages 711–716. IEEE, 2002.
17. H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. *Handbook of metaheuristics*, pages 320–353, 2003.
18. G. Minella, R. Ruiz, and M. Ciavotta. A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451, 2010.
19. N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.
20. R. S. Ottoni, J. E. C. Arroyo, and A. G. Santos. Algoritmo vns multi-objetivo para um problema de programação de tarefas em uma máquina com janelas de entrega. In *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, pages 1801 – 1812, 2011.
21. L. Tang and X. Wang. Simultaneously scheduling multiple turns for steel color-coating production. *European Journal of Operational Research*, 198(3):715–725, 2009.
22. J. Valente. An analysis of the importance of appropriate tie breaking rules in dispatch heuristics. *Pesquisa Operacional*, 26(1):169–180, 2006.
23. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.
24. E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms-a comparative case study. In *Parallel problem solving from nature-PPSN V*, pages 292–301. Springer, 1998.
25. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.